Autonomous Operation for Last-Mile Delivery on Suburban Sidewalk

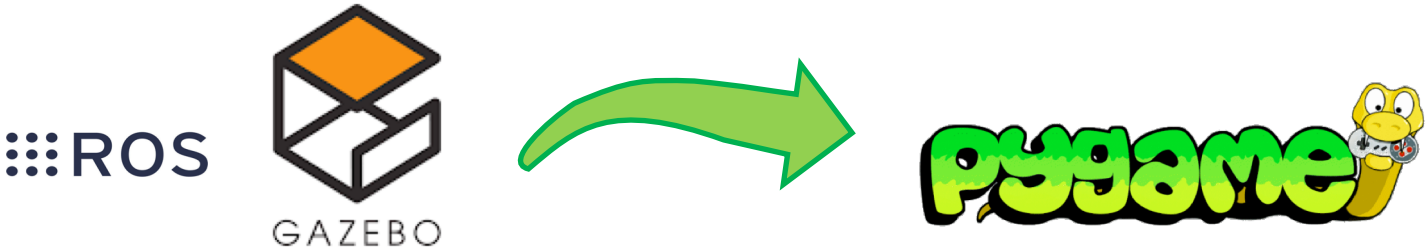
- RBE550-Project Update
- Keith Chester/Bob DeMont

A Couple of Changes

“No plan of operations extends with any certainty beyond the first encounter with the main enemy forces.”

-Prussian Field Marshal Helmuth von Moltke the Elder, 1871

With the extensive learning in Pygame for the homeworks and relatively little time left for learning ROS/Gazebo, we changed tools.



ROS/Gazebo out, Pygame in.

Goals

Our goals came into focus over the course of the lessons as we explored various search techniques.

Original Goals

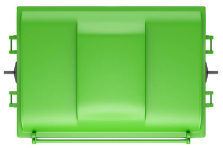
- Planning and execution algorithms
  - Autonomous navigation
  - Obstacle avoidance
    - Static obstacles (boxes, trashcans)
    - Dynamic obstacles (people bikes)
  - Sidewalk rules
    - Staying on sidewalk
    - Crossing at crosswalk at green light
  - Simulation visualization in Gazebo

Revised Goals

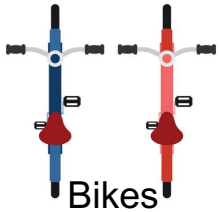
- Planning and execution algorithms
  - Autonomous navigation
    - Global planner for high level navigation
    - Local planner for kinematic movement and obstacle avoidance
  - Obstacle avoidance
    - Static obstacles (boxes, trashcans)
    - Dynamic obstacles (cars)
  - Sidewalk rules
    - Staying on sidewalk for cars
  - Simulation visualization in Pygame

Our Environment

Cones



Dumpsters



Bikes

Vehicles

Trashcans

Our Hero

Our delivery bot (Delbot) didn’t scale well but he’s there in sprite – I mean spirit!

Our Hero's Kinematics

- Diwheel kinematics

$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta$$

$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta$$

$$\dot{\theta} = \frac{r}{L}(u_r - u_l).$$

Approach

- Global planner for the shortest path/local planner for movement
- All nav aids/waypoints mapped and classified as corners, crosswalks and delivery addresses.
    - Known navigation map
    - Permitting an A* optimal search for shortest route to a delivery address

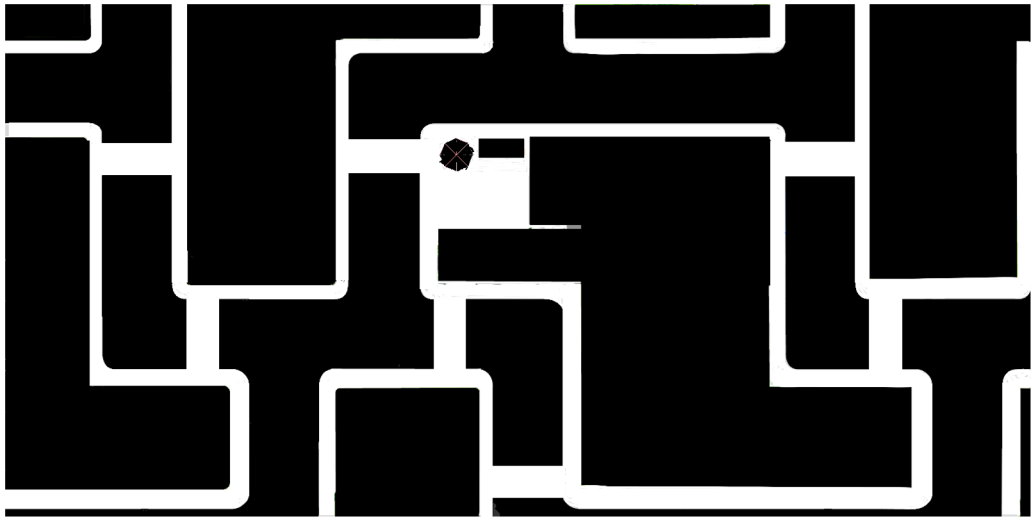
Here our hero is ready at the store near the center

For testing: Crosswalks and corners marked in black

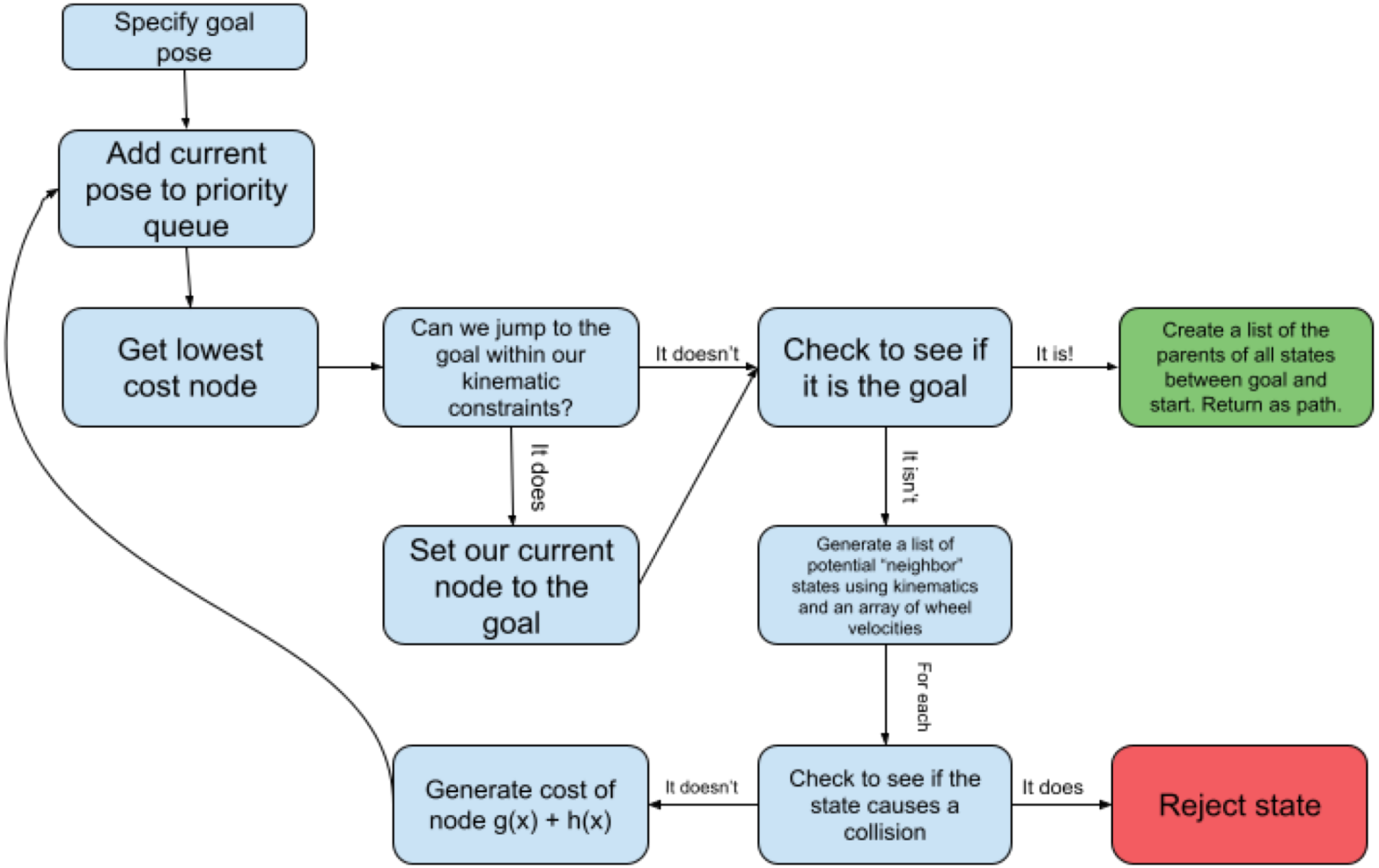
Delivery addresses in blue

Approach(cont)

- After the global planner determines the best path the local planner executes the movement.
- We modeled diwheel kinematics to execute movement.
- Collisions were detected on a shadow layer of the game using Pygame sprites and alpha layers (obstacles and their masks randomly inserted for each run)



A* with Kinematic Constraints

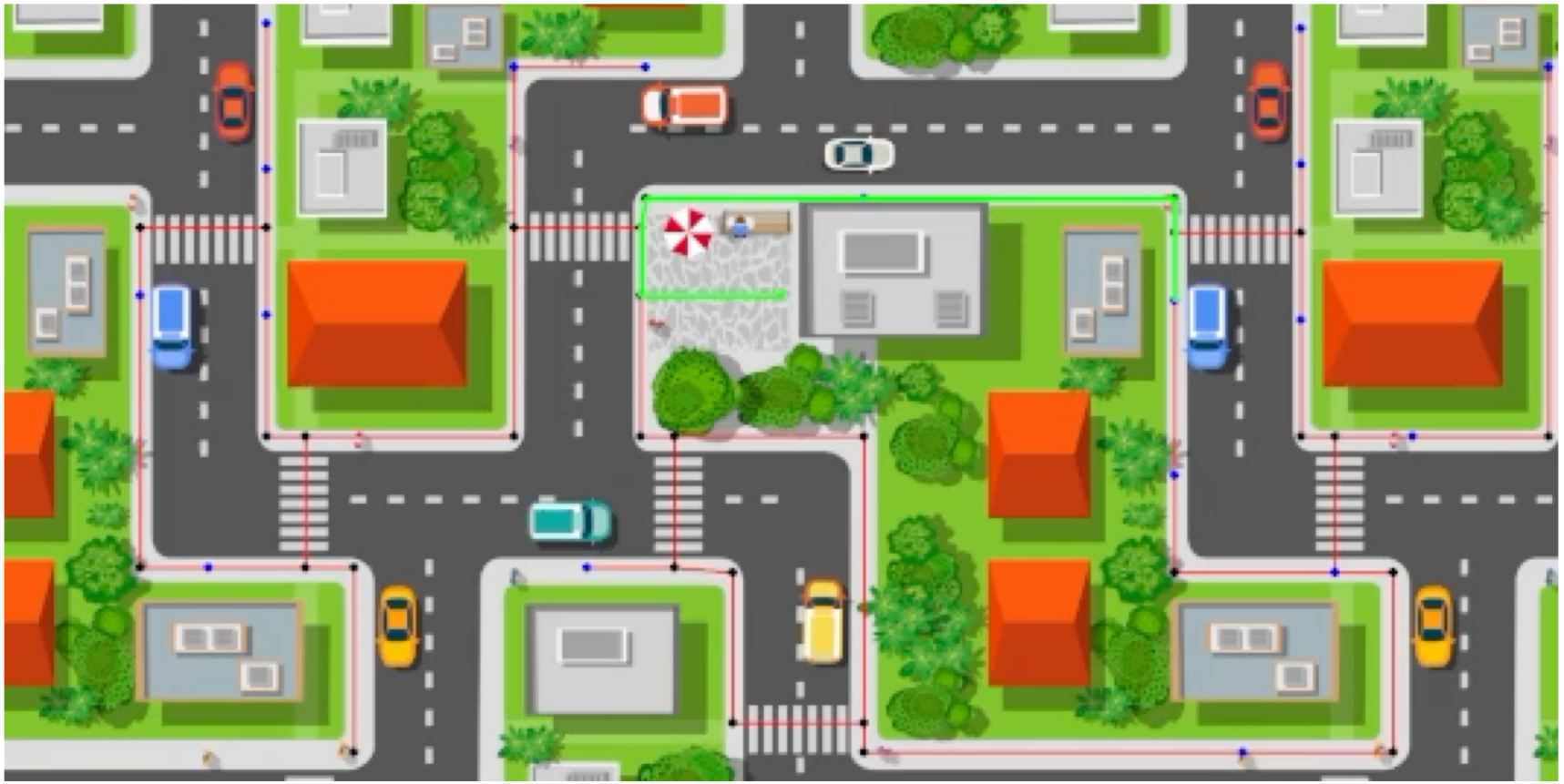


Results

We currently have :

- the environment complete
- Collisions detection methods working
- Random delivery addresses generating
- Global planning selecting optimized route
- Local planner working

Global Planner Example



Local Planner Example

More Exploration

Over the rest of the week we plan to:

- Complete debugging and tuning of the local planner
- Add randomized environmental obstacles
  - -To the visual environment
  - -as well as the collision environment
- Generate moving obstacles (cars)
- Complete written report